44장 REST API

- HTTP의 장점을 최대한 활용할 수 있는 아키텍처로서 REST가 소개되었고,

  이는 HTTP 프로토콜을 의도에 맞게 디자인하도록 유도하고 있다.
- REST의 기본 원칙을 성실히 지킨 서비스 디자인을 **RESTful** 이라고 표현한다.
  - 즉, REST는 HTTP를 기반으로 클라이언트가 서버의 리소스에 접근하는 방식을 규정한

    아키텍처고, REST API는 REST를 기반으로 서비스 API를 구현한 것을 의미

44.1 REST API의 구성

- REST API는 자원, 행위, 표현의 3가지 요소로 구성
- REST는 자체 표현 구조로 구성되어 REST API만으로 HTTP 요청의 내용을 이해할 수 있다.
  - **자원** : 자원 (URI)
  - **행위** : 자원에 대한 행위 (HTTP 요청 메서드)
  - **표현** : 자원에 대한 행위의 구체적 내용 (페이로드)

44.2 REST API 설계 원칙

- REST에서 가장 중요한 기본적인 원칙
  - URI는 리소스를 표현해야 한다.
  - 리소스에 대한 행위는 HTTP 요청 메서드로 표현한다.

1. URI는 리소스를 표현해야 한다.

- 리소스를 식별할 수 있는 이름은 동사보다는 명사를 사용한다.
- 따라서 이름에 get 같은 행위에 대한 표현이 들어가서는 안 된다.

```
# bad
GET /getTodos/1
GET /todos/show/1

# good
GET todos/1
```

2. 리소스에 대한 행위는 HTTP 요청 메서드로 표현한다.

- HTTP 요청 메서드는 클라이언트가 서버에게 요청의 종류와 목적을 알리는 방법이다.
- 주로 5가지 요청 메서드를 사용하여 CRUD를 구현한다.

HTTP 요청 메서드	종류	목적	페이로드
GET	index/retrieve	모든/특정 리소스 취득	X
POST	create	리소스 생성	○
PUT	replace	리소스의 전체 교체	○
PATCH	modify	리소스의 일부 수정	○
DELETE	delete	모든/특정 리소스 삭제	X

```
# bad
GET /todos/delete/1

# good
DELETE todos/1
```

44.3 JSON Server를 이용한 REST API 실습

JSON Server를 사용해 가상 REST API 서버를 구축하여 HTTP 요청을 전송하고 응답을 받을 수 있다.

44.3.4 GET 요청

- todos 리소스에서 모든 todo를 취득한다.
- public 폴더에 get_index.html 을 추가하고

  브라우저에서 http://localhost:3000/get_index.html로 접속한다.

```
<!DOCTYPE html>
<html>
<body>
    <pre></pre>
    <script>
        // XMLHttpRequest 객체 생성
    const xhr = new XMLHttpRequest();

        // HTTP 요청 초기화
    // todos 리소스에서 모든 todo를 취득(index)
        xhr.open('GET', '/todos');

    // todos 리소스에서 id를 사용하여 특정 todo 취득(retrieve)
    xhr.open('GET', '/todos/1');

        // HTTP 요청 전송
        xhr.send();

        // load 이벤트는 요청이 성공적으로 완료된 경우 발생한다.
        xhr.onload = () => {
        // status 프로퍼티 값이 200이면 정상적으로 응답된 상태다.
        if(xhr.response === 200){
            document.querySelector('pre').textContent = xh
r.response;
        }else{
            console.log('Error', xhr.status, xhr.statusTex
t);
        }
        };
  </script>
</body>
</html>
```

44.3.5 POST 요청

- todos 리소스에 새로운 todo를 생성한다.
- post 요청 시에는 setRequestHeader 메서드를 사용하여 요청 몸체에 담아 서버로 전송할 페이로드의 MIME 타입을 지정해야 한다.
- public 폴더에 다음 post.html을 추가하고 브라우저에서 http://localhost:3000/post_index.html로 접속한다.

```
<!DOCTYPE html>
<html>
<body>
    <pre></pre>
    <script>
    const xhr = new XMLHttpRequest();

        // HTTP 요청 초기화
        // todos 리소스에 새로운 todo를 생성
        xhr.open('POST', '/todos');

        // 요청 몸체에 담아 서버로 전송한 페이로드의 MIME 타입을 지정
        xhr.setRequestHeader('content-type', 'application/j
son');

        // HTTP 요청 전송
        // 새로운 todo를 생성하기 위해 페이로드를 서버에 전송해야 한
다.
        xhr.send(JSON.stringify({id: 4, content: 'Angular',
completed: false}));

        xhr.onload = () => {
            if(xhr.status === 200){
            document.querySelector('pre').textContent = xh
r.response;
        }else{
            console.log('Error', xhr.status, xhr.statusTex
t);
        }
```

```
        };
    </script>
</body>
</html>
```

44.3.6 PUT 요청

- PUT 요청 시에는 setRequestHeader 메서드를 사용하여 요청 몸체에 담아 서버로 전송할 페이로드의 MIME 타입을 지정해야 한다.
- public 폴더에 다음 put.html을 추가하고 브라우저에서 http://localhost:3000/put_index.html로 접속한다.

```
<!DOCTYPE html>
<html>
<body>
    <pre></pre>
    <script>
    const xhr = new XMLHttpRequest();

        // HTTP 요청 초기화
        // todos 리소스에서 id로 todo를 특정하여 id를 제외한 리소스 전체를 교체
        xhr.open('PUT', '/todos/4');

        // 요청 몸체에 담아 서버로 전송한 페이로드의 MIME 타입을 지정
        xhr.setRequestHeader('content-type', 'application/json');

        // HTTP 요청 전송
        // 리소스 전체를 교체하기 위해 페이로드를 서버에 전송해야 한다.
        xhr.send(JSON.stringify({id: 4, content: 'React', completed: false}));

        xhr.onload = () => {
        if(xhr.status === 200){
            document.querySelector('pre').textContent = xh
```

```
r.response;
        }else{
            console.log('Error', xhr.status, xhr.statusTex
t);
        }
        };
  </script>
</body>
</html>
```

44.3.7 PATCH 요청

- PATCT 요청 시에는 setRequestHeader 메서드를 사용하여 요청 몸체에 담아 서버로 전송할 페이로드의 MIME 타입을 지정해야 한다.
- public 폴더에 다음 patch.html을 추가하고 브라우저에서 http://localhost:3000/patch_index.html로 접속한다.

```
<!DOCTYPE html>
<html>
<body>
    <pre></pre>
    <script>
    const xhr = new XMLHttpRequest();

        // HTTP 요청 초기화
        // todos 리소스의 id로 todo를 특정하여 completed만 수정
        xhr.open('PATCH', '/todos/4');

        // 요청 몸체에 담아 서버로 전송한 페이로드의 MIME 타입을 지정
          xhr.setRequestHeader('content-type', 'applicatio
n/json');

        // HTTP 요청 전송
        xhr.send(JSON.stringify({completed: true}));

        xhr.onload = () => {
            if(xhr.status === 200){
```

```
            document.querySelector('pre').textContent = xh
r.response;
        }else{
            console.log('Error', xhr.status, xhr.statusTex
t);
        }
        };
  </script>
</body>
</html>
```

44.3.8 DELETE 요청

- public 폴더에 다음 delete.html을 추가하고 브라우저에서 http://localhost:3000/delete_index.html로 접속한다.

```
<!DOCTYPE html>
<html>
<body>
    <pre></pre>
    <script>
    const xhr = new XMLHttpRequest();

        // HTTP 요청 초기화
        // todos 리소스에서 id를 사용하여 todo를 삭제한다.
        xhr.open('DELETE', '/todos/4');

        // HTTP 요청 전송
        xhr.send();

        xhr.onload = () => {
            if(xhr.status === 200){
            document.querySelector('pre').textContent = xh
r.response;
        }else{
            console.log('Error', xhr.status, xhr.statusTex
t);
        }
```

```
        };
    </script>
</body>
</html>
```